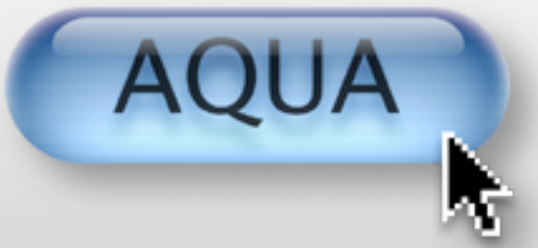
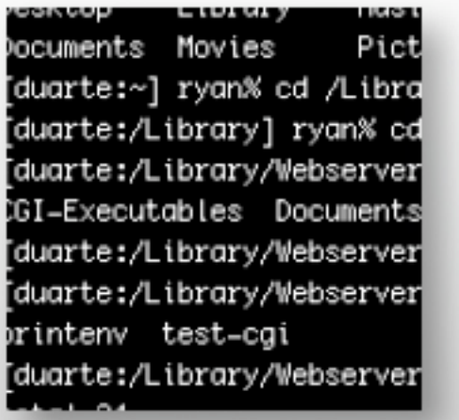
Cocoa User Interface Programming

Session 412





Cocoa User Interface Programming

Andreas Wendker
Manager, WebObjects Engineering
wendker@apple.com

Introduction

- Cocoa is the fastest and best way to develop desktop applications for Mac OS X…
- …and we are going to make it even better!

Introduction

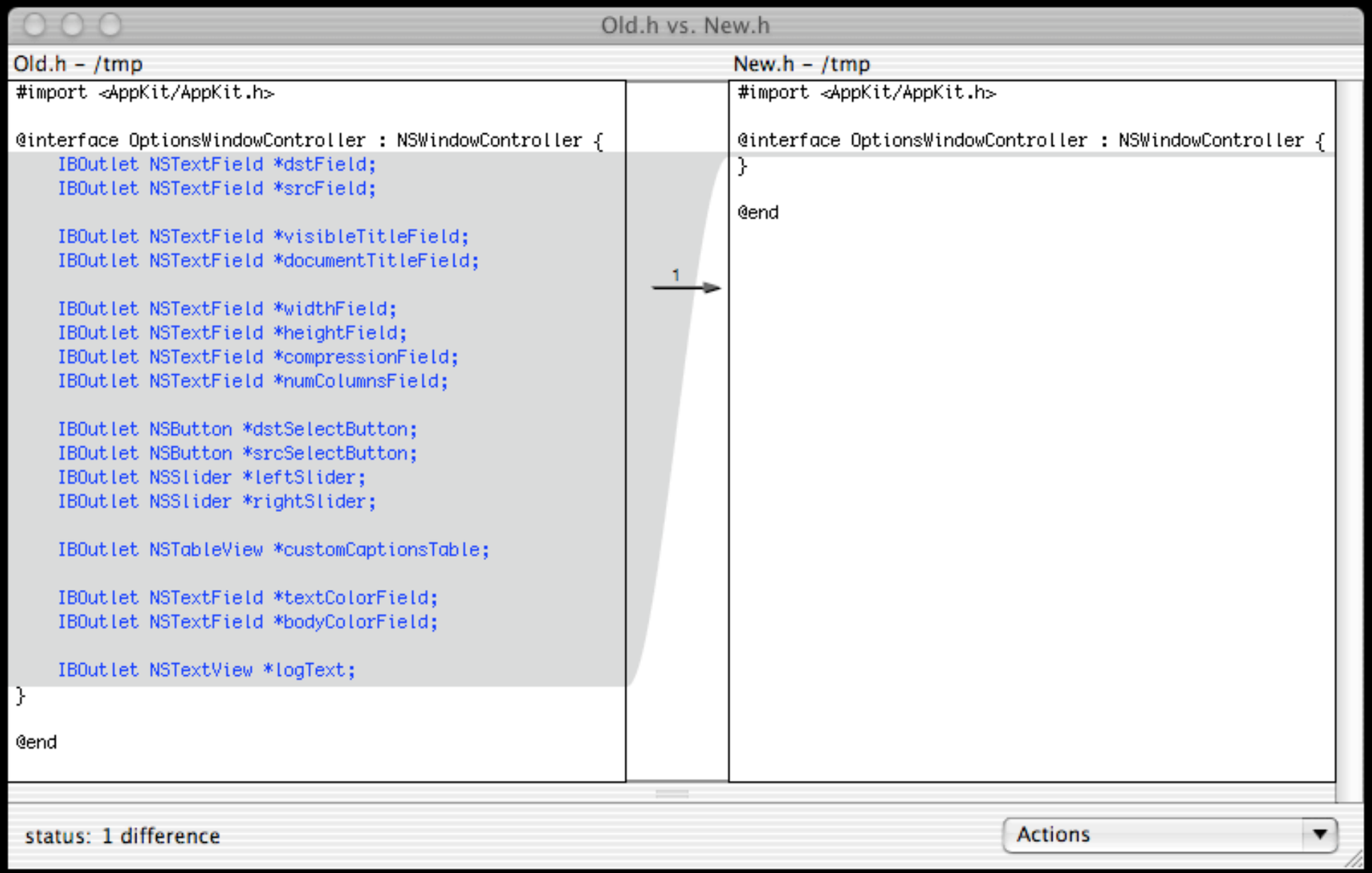
- A lot of code deals with keeping the user interface in sync with the data in underlying objects
  - Repetitive glue code
  - Many outlets, delegates
  - Routine work

Introduction

- In Panther we are going to introduce a complete new layer of functionality—a controller layer
  - Automates user interface handling
  - Many features you have been asking for
  - Interface Builder extended to support new features without writing code
  - Makes your development faster and easier

Introduction

- Effect on your applications: less glue code



What You'll Learn

- An overview of how the new API works
- Concepts build upon the well-known model-view-controller (MVC) paradigm
- How to use Interface Builder to configure the automatic synchronization of model objects and views

Demo

James Dempsey
Operations Engineer, Apple

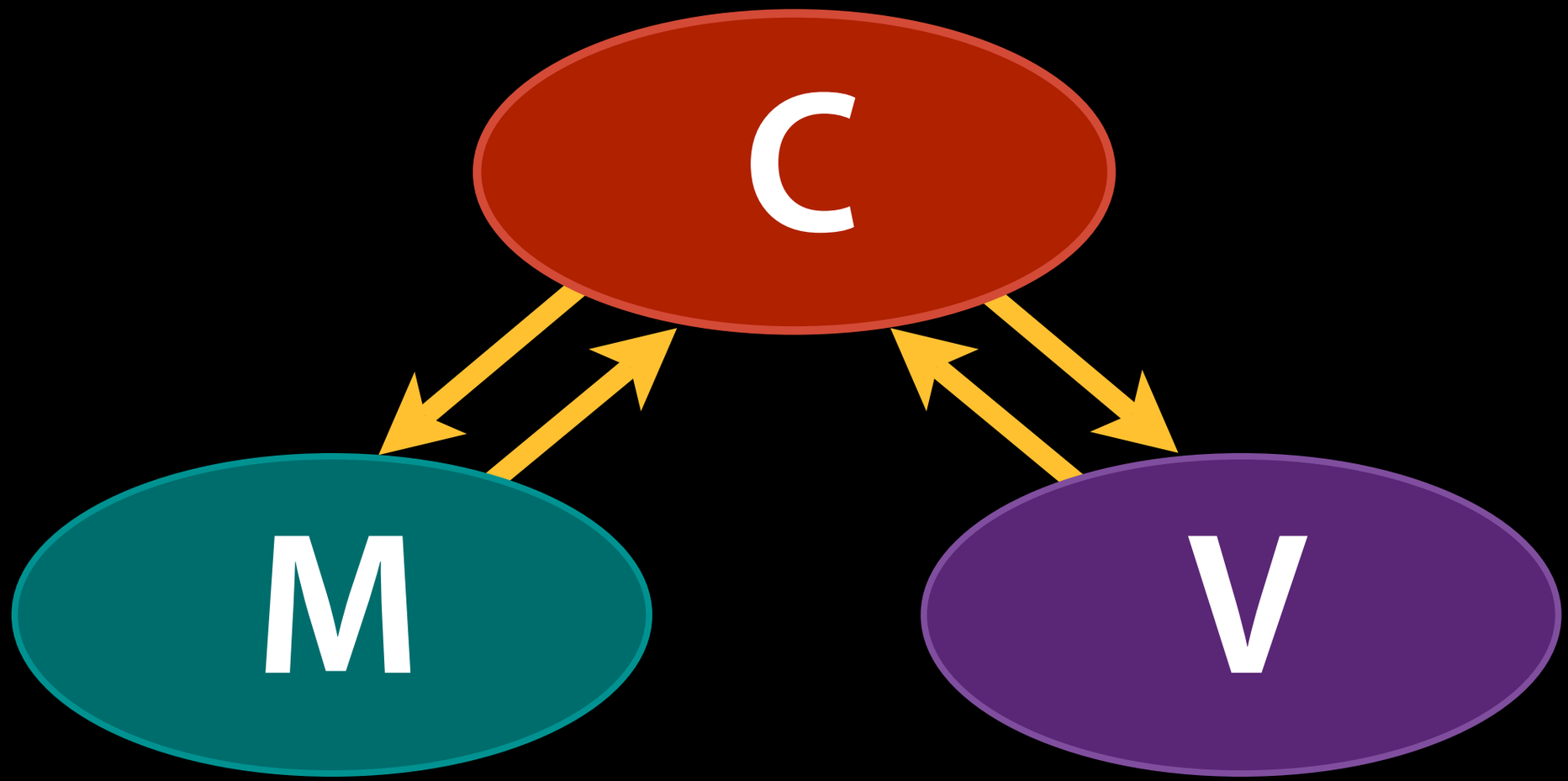
http://developer.apple.com/wwdc2003/urls.html

Our Goals for the Controller Layer

- Support a large variety of use cases and applications
- Integrate easily into existing applications
- Flexible and extensible
- Fast
- Great Interface Builder integration
- Easy to learn

Architecture

Model-View-Controller Paradigm



Features

Use any model object

- Apply new functionality to any model
  - NSDocuments
  - User defaults
  - NSDictionaries as generic containers
  - Custom model objects
- Common access protocol: Key Value Coding (KVC)

Features

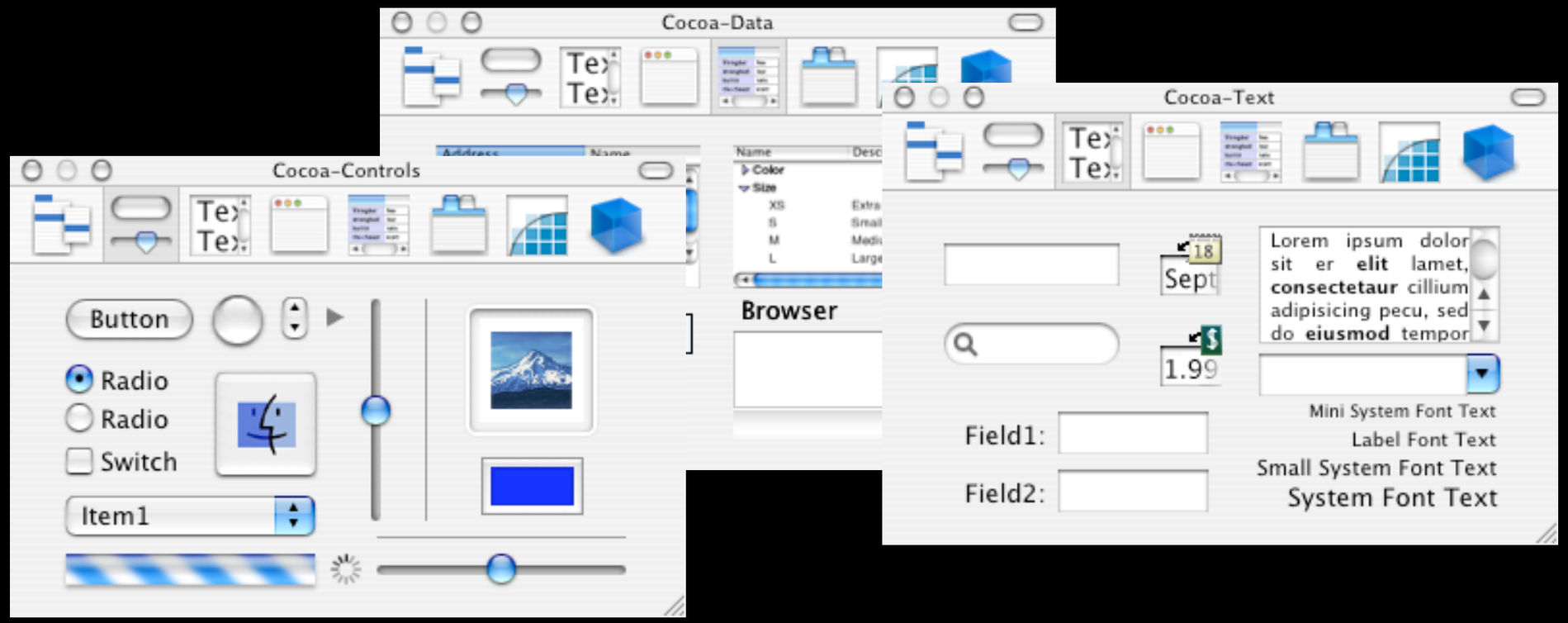
Automatic user interface refresh

- Different parts of the user interface synchronized automatically
  - Multiple document windows stay in sync
- Common observing protocol: Key Value Observing (KVO)
- Automatic propagation of changes
  - Changes through Scripting immediately visible
  - Preferences applied immediately

Features

Display and edit all standard Cocoa data types

- Strings, numbers/booleans, dates, colors, images, fonts, paths, URLs
- Bindings for nearly all Cocoa controls
- Tables for object arrays



Features

Manage selections

- Dependencies in user interface handled automatically
- Automatically synchronize widget content with selection
- Editing of multiple selected objects at the same time

Features

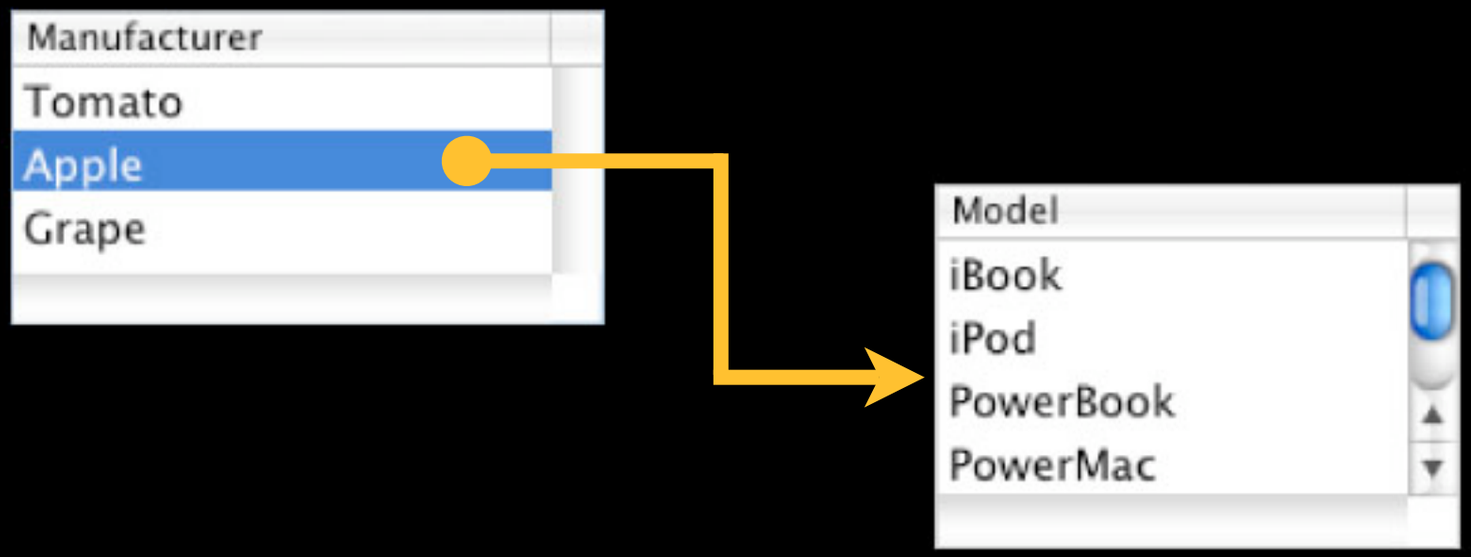
Common user interface states handled cleanly

- Informative user interfaces
- Marker values for the following states:
  - Null value (nil)
  - Multiple values selection
  - No selection
  - Not applicable
- Define placeholder values for each state as needed/desired

Features

Manipulate object relationships

- Manage references of model objects to each other
- Arrays (“to-many”) or single objects (“to-one”)
- Easily traverse multiple relationships



Features

Value Transformers for customizations

- Convert values before displayed, after being edited
- Apply “expressions” to values
- Default transformers
  - Negate booleans
  - Test values for nil/not nil
  - Archive values into NSData

Features

Control display parameters

- User friendly applications made simple
- Automatic sorting in table views
- Manage large variety of user interface parameters, for example
  - Display colors and fonts
  - Window, table column sizes
  - Tab view selections
- Store in user defaults or per document
  - Shared user defaults controller accessible in nib files

Demo

James Dempsey
Operations Engineer, Apple

http://developer.apple.com/wwdc2003/urls.html

API

- Utility classes
- Controller classes
- Protocols

API

- Utility classes
  - NSIndexSet/NSMutableIndexSet
  - NSSortDescriptor
  - NSValueTransformer
- Controller classes
- Protocols

API

- Utility classes
- Controller classes
  - NSController
  - NSUserDefaultsController
  - NSObjectController
  - NSArrayController
- Protocols

API

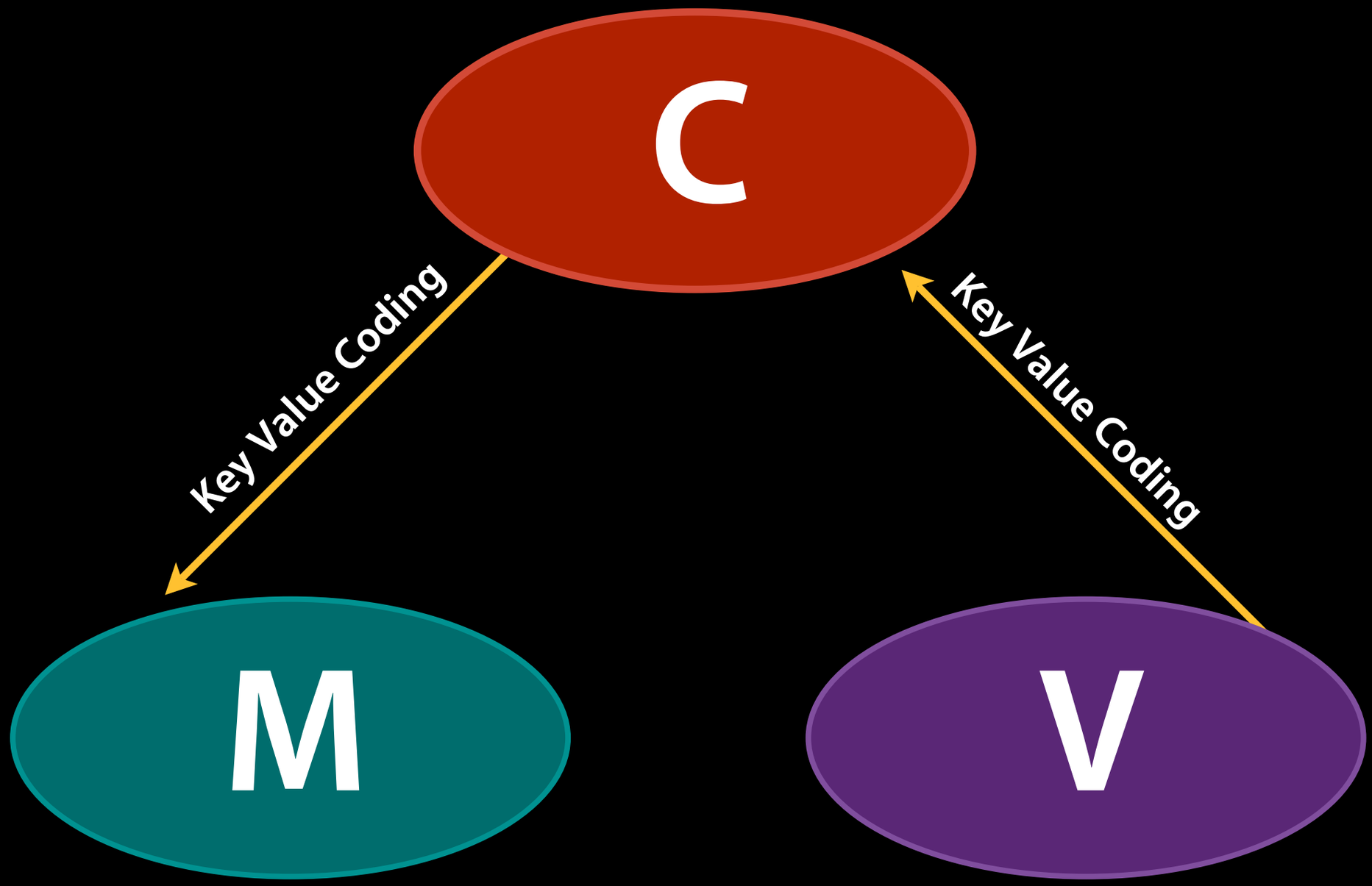
- Utility classes
- Controller classes
- Protocols
  - Key Value Coding
  - Key Value Observing
  - Editor Tracking

API

Protocols

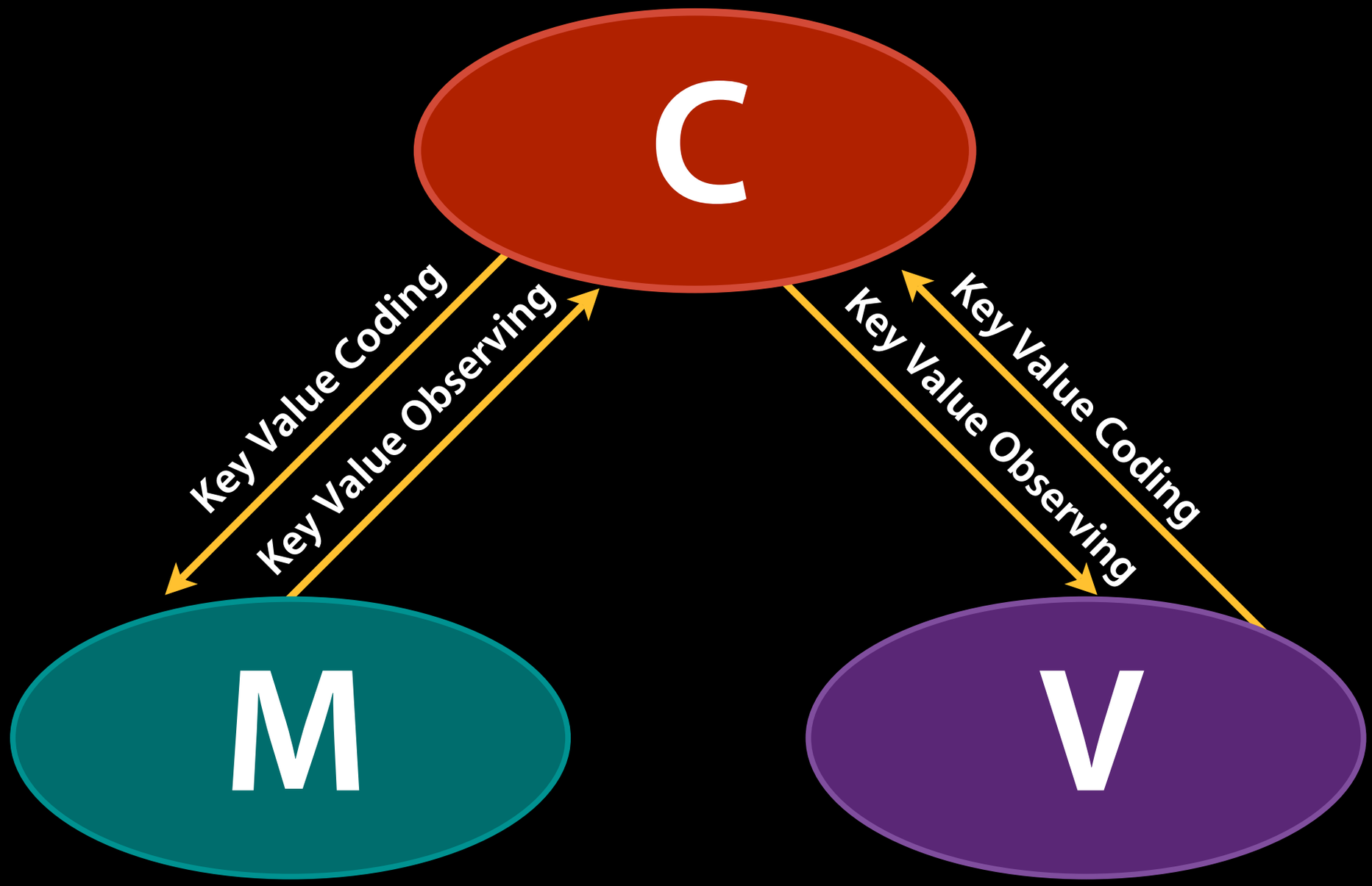
API

Protocols



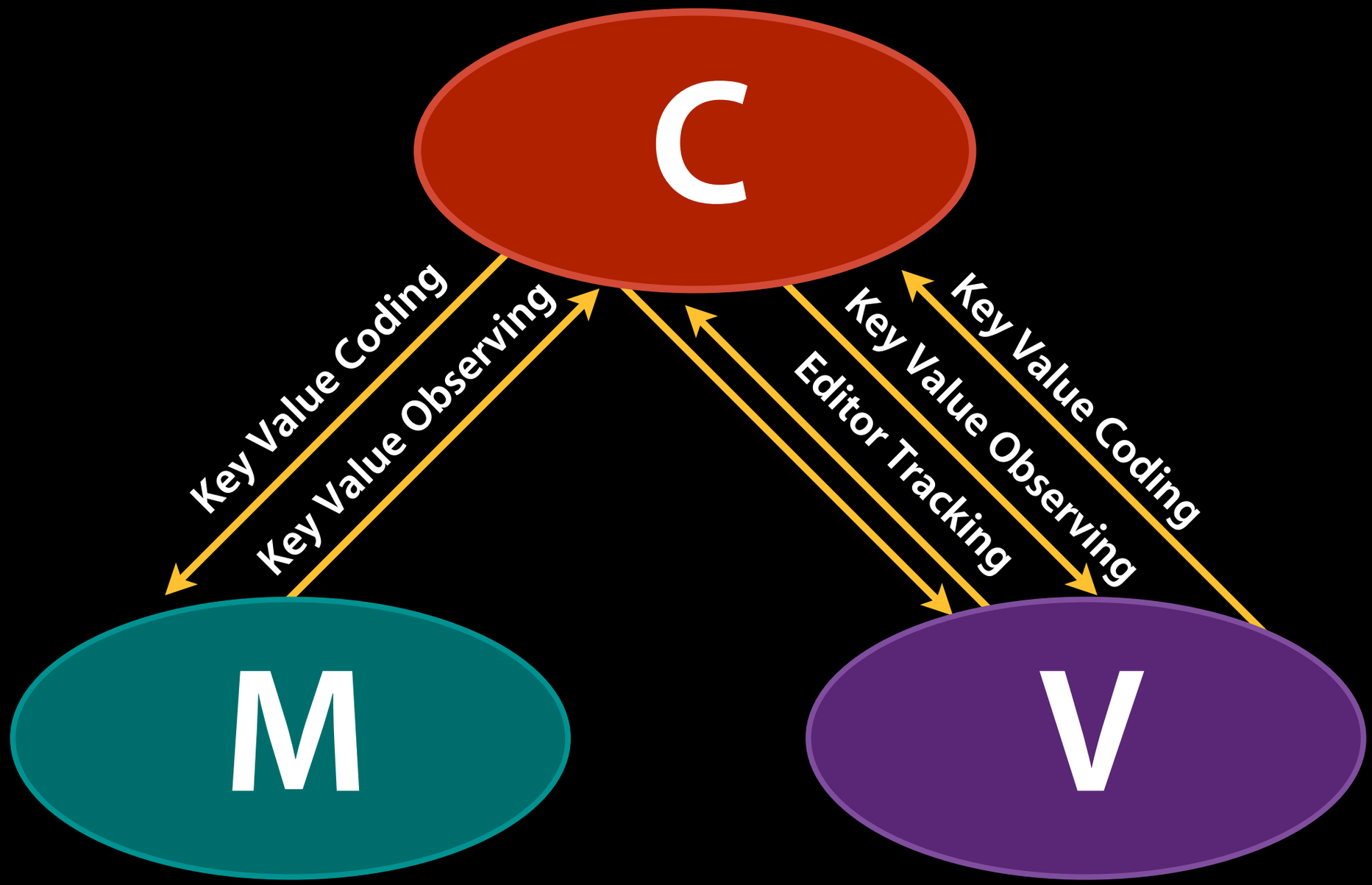
API

Protocols



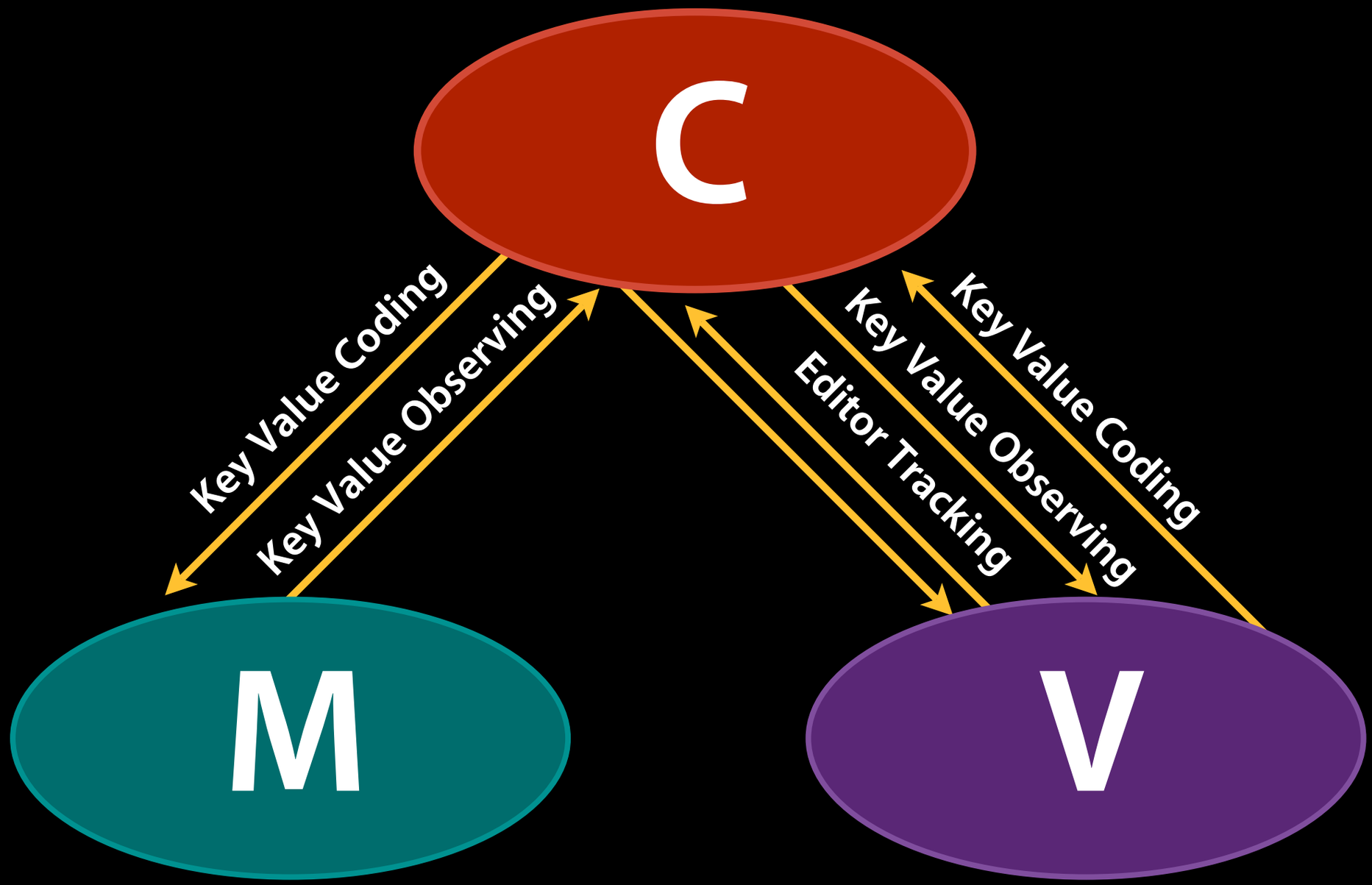
API

Protocols



API

Protocols



API: Configuring Bindings

- Creating bindings

```
- (void)bind:(NSString *)binding
    toObject:(id)observable
 withKeyPath:(NSString *)keyPath
     options:(NSDictionary *)options;

- (void)unbind:(NSString *)binding;
```

-

-

```
@”NSValueTransformerName”
@”NSNullPlaceholder”
@”NSNoSelectionPlaceholder”
@”NSMultipleValuesPlaceholder”
```

API: Setting Default Placeholders

- Defining default placeholder values per view class and binding

```
+ (void)setDefaultPlaceholder:(id)placeholder
                    forMarker:(id)marker
                  withBinding:(NSString *)binding;

+ (id)defaultPlaceholderForMarker:(id)marker
                      withBinding:(NSString *)binding;
```

- Markers

```
id NSMultipleValuesMarker;
id NSNoSelectionMarker;
id NSNotApplicableMarker;
```

API: Additional Bindings

- Exposing keys as bindings

```
+ (void)exposeBinding:(NSString *)binding;

- (NSArray *)exposedBindings;

// optional, support Interface Builder:
- (Class)valueClassForBinding:(NSString *)binding;
```

- Any key value coding compliant parameter can be exposed without addtional work
- Advanced bindings require overriding bind method

API: Populating Controllers

- NSObjectController, NSArrayController content

```
+ (void)setContent:(id)content;
- (id)content;
```

- Use bindings

```
objectRelationship
arrayRelationship
```

Demo

James Dempsey
Operations Engineer, Apple

http://developer.apple.com/wwdc2003/urls.html

Performance

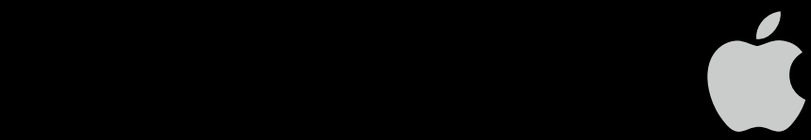
- User interface refresh is well tuned
  - Single cells in table views
  - Redrawing avoided through smart caching
- Key Value Coding adds a level of indirection
  - A little slower than direct method invocations
  - Use Key Value Coding the way it is intended

Integrating Into Existing Applications

- Using the new functionality does not require a full rewrite of your application
- Controller layer can easily co-exist with traditional glue code
- Start using it soon
  - Better user experience: Selection handling, sorting, placeholders, user defaults
  - Improved performance
  - Reduce maintenance cost
  - Be better prepared for future releases

Panther WWDC Seed

- Little documentation: Headers
- Some pieces missing
  - Controllers don’t use key value observing, yet
  - Available bindings, binding names preliminary
  - Targets, delegates, data sources are taken by value and content bindings
- Interface Builder: use “10.2 and later” nib file format for enhanced compatibility



Feedback Welcome

- Please submit bug reports
- Send feedback and thoughts:
  - cocoa-feedback@group.apple.com

Summary

- New controller layer introduces a new paradigm for developing Cocoa applications
  - Write less code, develop faster
  - Cleanly handle common selections states
  - Automatic sorting
- Value transformers facilitate easy customization
- Interface Builder extended to support new features

Song

James Dempsey
Operations Engineer, Apple

http://developer.apple.com/wwdc2003/urls.html

Wrap Up

Roadmap

400 Introduction to Developing Applications with Cocoa	Nob Hill Mon., 5:00pm
413 Cocoa 2D Drawing Techniques	Nob Hill Wed., 5:00pm
419 What's New in Cocoa User Interface Programming in Depth	Russian Hill Fri., 9:00 am
424 Cocoa Tips and Tricks	Mission Fri., 2:00 pm
427 What's New in Cocoa Text	Presidio Fri., 5:00 pm
FF008 Cocoa Feedback Forum	North Beach Thu., 10:30 am

Who to Contact

John Geleynse
Manager, Software Evangelism
geleynse@apple.com

Cocoa Feedback
For suggestions and feedback
cocoa-feedback@group.apple.com

Developer Technical Support
Code-level questions (fee-based)
dts@apple.com

For More Information

Reference Library

Documentation

Forthcoming—check headers
/System/Library/Frameworks/AppKit.framework/Headers

Demo application code *(WWDC-only distribution)*
ADC Member Site > Download Software > WWDC 2003

Q&A

Apple Software Engineers

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference